Методичка 8.4 - Обзор возможностей дизассемблера IDA Pro

Удаленная отладка

Сейчас мы немного поговорим об удаленной отладке приложений. Зачем вообще нужна удаленная отладка, если приложение можно отладить локально. Действительно, если есть возможность отладить приложение локально, то в большинстве случаев это будет самым удобным способом, но это не всегда возможно, особенно, когда мы хотим выполнить анализ в дизассемблере IDA Pro.

Например, необходимо отладить какой-нибудь модуль в составе прошивки IP-камеры или роутера, где скорее всего нас будет ждать процессор с архитектурой ARM. Как вы понимаете, IDA Pro не получится запустить на подобных устройствах.

В таких случаях запускают необходимый модуль на удаленном устройстве, например, под легковесным отладчиком gdbserver, который по сети получает команды, выполняет их и отправляет результат обратно также по сети.

gdbserver занимает очень мало места и его без труда можно найти в сети Интернет, практически, под любую архитектуру процессора, ну или собрать самостоятельно.

Я продемонстрирую, как может быть использована связка gdbserver и IDA Pro. Узнаем IP-адрес удаленной машины.

$ ifconfig

Запускаем приложение с использованием gdbserver.

$ gdbserver 0.0.0.0:1337 prog-6.4

Затем переходим в операционную систему Windows и запускаем IDA Pro. Выбираем Debug - Attach - 192.168.1.7:1337 - Ok.

И мы получаем код удаленного приложения в дизассемблере IDA Pro. В рамках курса вам это не делать не придется, но знать о такой возможности будет полезно.

Декомпилятор Hex-Rays

Помимо того, что дизассемблер IDA Pro умеет очень хорошо выполнять анализ дизассемблированного листинга, ее функциональность может расширена за счет декомпилятора Hex-Rays.

Декомпиляция - это процесс получения псевдокода на основе дизассмблированного листинга.

Я продемонстрию работу декомпилятора Hex-Rays на примере программ, анализ которых мы выполняли на протяжении курса.

Открываем программу prog-5.1. Переходим на функцию main и нажимаем F5. И мы получаем код, который максимально приближен к языку Си. Думаю, очевидно, что проводить анализ программы по псевдокоду гораздо быстрее, чем разбирать инструкции на языке Ассемблер.

Можем выбрать еще другую функцию и также выполнить декомпиляцию.

В рамках курса не подразумевается, что вы будете использовать декомпилятор для решения ДЗ, но знать о такой возможности будет однозначно полезно.

Анализ патча с помощью плагина BinDiff

Исходный код программы prog-6.4

...

Компиляция программы:

> cl prog-6.4.c -Od /Gs- /GS- /link /DYNAMICBASE:NO /NXCOMPAT:NO

В самом первом видео мы немного говорили о том, что поиск 1-day уязвимостей может быть выполнен путем анализа патчей.

Для демонстрации были подготовлены программы prog-6.4.exe и prog-6.4-fix.exe. Первая содержит уязвимость, а во второй она уже исправлена.

Давайте посмотрим, как выполняется сравнение двух версий программы. В этом нам поможет замечательный плагин, который называется BinDiff или Diaphora.

Выбираем оба файла.

И мы видим в левой части экрана исходный код программы prog-6.4, а в правой части исходный код исправленной версии программы.

Цветом выделен код, не совпадает. Видим, что пропатченная версия отличается наличием кода, который проверяет длину буфера перед вызовом функции strcpy.

Делаем вывод, что, скорее всего, в программе была исправлена уязвимость переполнения буфера.